COMP 1204 Coursework 2

Name: Palak Jain

Student ID: 30012627

11/03/2019

1 The Relational Model

1.1 EX1:

HotelReviews(String: HotelID, String: HotelURL, Integer: OverallRating, Float: Avg.Price, String: Author, Text: Content, Text: Date, Integer: No. Reader, Integer: No. Helpful, Integer: Overall, Integer: Value, Integer: Rooms, Integer: Location, Integer: Cleanliness, Integer: CheckIn/Frontdesk, Integer: Service, Integer: Business Service)

Primary key: HotelID, Author, Date

1.2 EX2:

Functional Dependencies: HotelID, Author, Date functionally determine all other attributes.
Candidate Key: (HotelID, Author, Date)
Explanation: an author can only give one review on a particular day. Thus, these three attributes differentiate any review from another.

1.3 EX3:

1NF: None of the columns have multiple values. Thus 1NF is satisfied.

2NF: No non-prime attribute should be dependent on the proper subset of any candidate key of table. The candidate key is: HotelID, Author, Date. We know that HotelURL, OverallRating, Avg.Price are all dependent on HotelID. Additionally, Author is dependent on AuthorID.
Thus, we have 3 tables in the 2NF form:

Hotel(HotelID, Hotel URL, OverallRating, Avg. Price)
Primary key = HotelID

Reviews(HotelID, AuthorID, Content, Date,No. Readers, No. Helpful, Overall,Value, Rooms, Location, Cleanliness, Check In/ front desk,Service, Business service)
Primary key = HotelID,AuthorID, date
Foreign key from Hotel relation = HotelID
Foreign key from Author relation = AuthorID

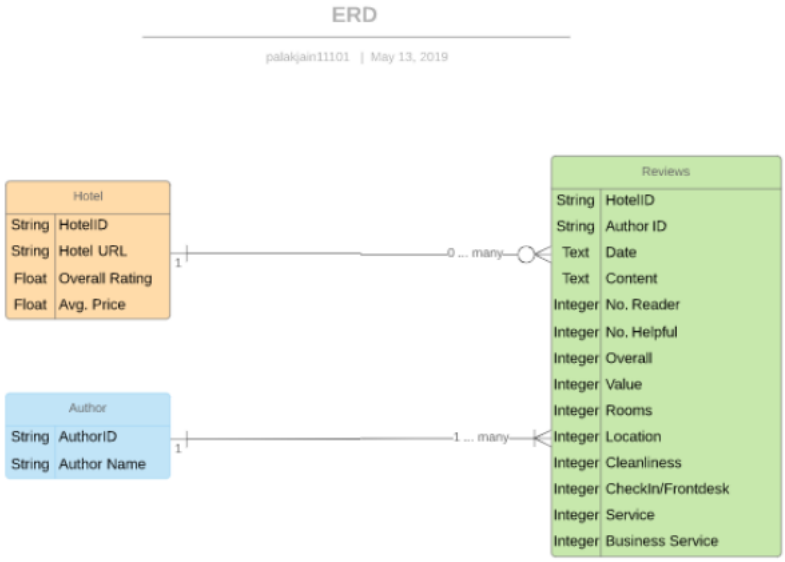
Author(Author ID, Author Name)
Primary key = AuthorID

3NF: Transitive functional dependency of non-prime attribute on any super key should be removed. No such transitive functional dependencies exist therefore current tables satisfy 3NF.

BCNF: For every functional dependency X $\rightarrow$ Y, X should be the super key of the table. In each of our three tables, the respective primary keys functionally determine all other attributes. Thus, BCNF is satisfied, and the tables cannot be normalized further.

2 Entity-Relationship Diagramming

2.1 EX4:



3 Relational Algebra

EX5:
$\sigma_{AuthorID=<ID>}(Reviews)$

EX6:
$(_{Author}\ \gamma_{\ count(hotelID)\rightarrow hotelIDcount}$
$(\sigma_{count>2}(_{Author,\ HotelID}\ \gamma_{\ count(Author)\rightarrow reviewsCount})))(Reviews)$

EX7:
$\Pi_{HotelID}(\sigma_{count>10}(_{HotelID}\ \gamma_{\ count(HotelID)\rightarrow count}))(Reviews)$

EX8:
$\Pi_{HotelID}(\sigma_{OverallRating>3\ \wedge\ AvgCleanliness>=4.5}$
$(_{HotelID,OverallRating}\ \gamma_{\ avg(CleanlinessRating)\rightarrow AvgCleanliness}))(Reviews \bowtie Hotel)$

4 SQL Queries

4.1 EX9:

sqlite3 main.db

CREATE TABLE HotelReviews(HotelID STRING, HotelURL STRING, OverallRating FLOAT, Avg.Price FLOAT, AuthorID STRING, Author STRING, Content TEXT, Date TEXT, No.Reader INTEGER, No.Helpful INTEGER, Overall INTEGER, Value INTEGER, Rooms INTEGER, Location INTEGER, Cleanliness INTEGER, CheckIn/Frontdesk INTEGER, Service INTEGER, BusinessService INTEGER, PRIMARY KEY(Author, Date, HotelID));

4.2 EX10:

The bash script is attached in the appendix of the report.

4.3 EX11:

CREATE TABLE Reviews(HotelID STRING, AuthorID STRING, Author STRING, Content TEXT, Date DATE, No.Reader INTEGER, No.Helpful INTEGER, Overall INTEGER, Value INTEGER, Rooms INTEGER, Location INTEGER, Cleanliness INTEGER, CheckIn/Frontdesk INTEGER, Service INTEGER, BusinessService INTEGER, PRIMARY KEY(Author, Date, HotelID));

CREATE TABLE Hotel(HotelID STRING, HotelURL STRING, OverallRating FLOAT, Avg.Price FLOAT, PRIMARY KEY(HotelID));

CREATE TABLE Author(AuthorID STRING, Author STRING, PRIMARY KEY(AuthorID));

4.4 EX12:

INSERT INTO Hotel (HotelID, HotelURL, OverallRating, Avg.Price)
SELECT HotelID, HotelURL, OverallRating, Avg.Price
FROM HotelReviews;

INSERT INTO Reviews(HotelID, AuthorID, Author, Content, Date, No.Reader, No.Helpful, Overall, Value, Rooms, Location, Cleanliness, CheckIn/Frontdesk, Service, BusinessService)
SELECT HotelID, HotelURL, OverallRating, Avg.Price, AuthorID, Author, Content, Date, No.Reader, No.Helpful, Overall, Value, Rooms, Location, Cleanliness, CheckIn/Frontdesk, Service, BusinessService
FROM HotelReviews;

INSERT INTO Author (AuthorID, Author)
SELECT AuthorID, Author
FROM HotelReviews;

DROP TABLE HotelReviews IF EXISTS;

4.5 EX13:

An index is a schema object that is used to quicken the speed of data retrieval of rows by using a pointer. Using an index can reduce disk I/O and helps to locate data more easily. The primary key of each table already makes the lookup quickly, so there is no need for it to be indexed. Three attributes make up the primary key for the Reviews table, all of them should separately have indexes. Having

an index for HotelID would make it quicker to access all the reviews associated with a particular hotel. Similarly, Date and Author ID should also have indexes, as look up for a specific Author, or reviews on a particular date would become easier. Author in the Author table and HotelURL in the Hotel table too, should be indexed to increase the speed of searches for values in these attributes.

```
CREATE INDEX hotelID_index
    ON Reviews(HotelID);

CREATE INDEX authorID_index
    ON Reviews(AuthorID);

CREATE INDEX date_index
    ON Reviews(Date);

CREATE INDEX author_index
    ON Author(Author);

CREATE INDEX url_index
    ON Hotel(HotelURL);
```

4.6 EX14:

SQL versions:

EX5:

```
SELECT *
FROM Reviews
WHERE Author =
```

EX6:

```
INSERT INTO Newtable (Author, HotelID, COUNT(Author))
SELECT Author, HotelID, COUNT(Author) AS reviewsCount
FROM Reviews
GROUP BY Author, HotelID
WHERE reviewsCount > 2;

SELECT Author, COUNT(HotelID)
FROM Newtable
GROUP BY Author
```

EX7:

```
SELECT HotelID
FROM Reviews
GROUP BY HotelID
WHERE COUNT(HotelID) > 10;
```

EX8:

```
SELECT Reviews.HotelID
```

FROM Reviews, Hotel
GROUP BY Reviews.HotelID, OverallRating
WHERE Reviews.HotelID = Hotel.HotelID AND OverallRating > 3 AND AVG(Cleanliness) > 4.5

5 Conclusions

5.1 EX15:

Although the bash script attached in the appendix has not taken into account some of the problems discussed below, how these problems could have been handled is explained.

Reviewers may have the same name and therefore it is important to differentiate them by giving every user a AuthorID(which will be unique for 2 different people with the same name). This AuthorID attribute can be made to autoincrement, unless the same author has written more reviews. In this case, the AuthorID will be the same in all these reviews.

Some of the reviews had invalid data (e.g. ¡img, showReview(. . .)) and erroneous values such as -1, this could manipulate results such as the average values of fields for each hotel. Such data would be treated as null values as it can cause problems with averages and aggregates.

There is also whitespace at the start/end and empty strings. Using gsub command inside these awk statements should get rid of these extra spaces.

There is also an instance where an author posts more than one review on a particular hotel on tthe same date. This would break the primary key so I would use the 'INSERT OR REPLACE INTO' statement rather than just 'INSERT INTO' so that only the data for the newer review is inserted into the tables.

6 Appendix

Please find below the screenshots of the bash script. The script is also available in the same generatesql.sh along with all the other files in the archive.

```
#!/bin/bash
#telling the environment/os to use bash as a command interpreter
#touch hotelreviews.sql
echo -e "DROP TABLE IF EXISTS HotelReviews; CREATE TABLE HotelReviews(
        HotelID STRING,
        HotelURL STRING,
        OverallRating FLOAT,
        AvgPrice FLOAT,
        Author STRING,
        Content TEXT,
        Date TEXT,
        NoReader INTEGER,
        NoHelpful INTEGER,
        Overall INTEGER,
        Value INTEGER,
        Rooms INTEGER,
        Location INTEGER,
        Cleanliness INTEGER,
        CheckInFrontDesk INTEGER,
        Service INTEGER,
        BusinessService INTEGER,
        PRIMARY KEY(HotelID, AuthorID, Author)

);" >>hotelreviews.sql
```

```
#if an argument is given along with the script in the commandline
if [ $# -eq 1 ] && [ -d "$1" ]; then
        for file in $1/*; do #for every file in the folder
                hotelID=${file%.*} # remove extension
                hotelID=${hotelID##*/} # remove path
                OverallRating=$(awk '/Overall Rating/{sum = $2}' $file)
                URL=$(awk '/HotelURL/{sum = $2}' $file)
                Price=$(awk -F '>' '/<Avg. Price>/{sum = $2}END{printf (sum)}' $file)
                count=$(awk -F '>' '/<Author>/{sum +=1}END{print sum}' $file)

                for count in file; do
                Author=$(awk -F '>' '/<Author>/{sum = $2}END{printf (sum)}' $file)
                Content=$(awk -F '>' '/<Content>/{sum = $2}END{printf (sum)}' $file)
                Date=$(awk -F '>' '/<Date>/{sum = $2}END{printf (sum)}' $file)
                Reader=$(awk -F '>' '/<No. Reader>/{sum = $2}END{printf (sum)}' $file)
                Helpful=$(awk -F '>' '/<No. Helpful>/{sum = $2}END{printf (sum)}' $file)

                Overall=$(awk -F '>' '/<Overall>/{sum = $2}END{printf (sum)}' $file)
                Value=$(awk -F '>' '/<Value>/{sum = $2}END{printf (sum)}' $file)
                Rooms=$(awk -F '>' '/<Rooms>/{sum = $2}END{printf (sum)}' $file)
                Location=$(awk -F '>' '/<Location>/{sum = $2}END{printf (sum)}' $file)
                Cleanliness=$(awk -F '>' '/<Cleanliness>/{sum = $2}END{printf (sum)}' $file)
                CheckIn=$(awk -F '>' '/<Check In>/{sum = $2}END{printf (sum)}' $file)
                Service=$(awk -F '>' '/<Service>/{sum = $2}END{printf (sum)}' $file)
                Business_service=$(awk -F '>' '/<Business service>/{sum = $2}END{printf (sum)}' $file)
                echo -e "INSERT INTO HotelReviews values ($hotelID,$URL, $OverallRating,$Price, $Author, $Content, $Date, $Reader, $Helpful,
                $Overall, $Value, $Rooms, $Location, $Cleanliness, $CheckIn, $Service, $BusinessService)" >> hotelreviews.sql
                #echo $Author $hotelID

        done
        #sort average ratings in descending order
        done
else
        #if absolute/relative path is not provided in the
        #command line, inform the user that it must be provided
        echo "Please provide the path for `basename $0`"
fi
#EOF
~
```